A Description Language for Syntactically Annotated Corpora

Esther König and **Wolfgang Lezius**
IMS, University of Stuttgart, Germany
www.ims.uni-stuttgart.de/projekte/TIGER

Abstract

This paper introduces a description language for syntactically annotated corpora which allows for encoding both the syntactic annotation to a corpus and the queries to a syntactically annotated corpus.

In terms of descriptive adequacy and computational efficiency, the description language is a compromise between script-like corpus query languages and high-level, typed unification-based grammar formalisms.

1 Introduction

Syntactically annotated corpora like the Penn Treebank (Marcus et al., 1993), the NeGra corpus (Skut et al., 1998) or the statistically disambiguated parses in (Beil et al., 1999) provide a wealth of information, which can only be exploited with an adequate query language. For example, one might want to retrieve verbs with their sentential complements, or specific fronting or extraposition phenomena. So far, queries to a treebank have been formulated in scripting languages like tgrep, Perl or others. Recently, some powerful query languages have been developed: an example of a high-level, constraint-based language is described in (Duchier and Niehren, 1999). (Bird et al., 2000) propose a query language for the general concept of annotation graphs. A graphical query notation for trees is under development in the ICE project (UCL, 2000).

In the current paper, we present a proposal for a graph description language which is meant to fulfill two conflicting requirements: On the one hand, the language should be close to traditional linguistic descriptions languages, i.e. to grammar formalisms, as a basis for modular, understandable code, even for complex corpus queries. On the other hand, the language should not preclude efficient query evaluation. Our answer is to profit from the research on typed, feature-based/constraint-based grammar formalisms (e.g. (Carpenter, 1992), (Copestake, 1999), (Dörre and Dorna, 1993), (Dörre et al., 1996), (Emele and Zajac, 1990), (Höhfeld and Smolka, 1988)), and to pick those ingredients which are known to be computationally 'tractable' in some sense.

2 The Query Language

2.1 The right kind of graphs

If syntactic analysis is meant to provide for a basis of semantic interpretation, the predicate-argument structure of a sentence must be recoverable from its syntactic analysis. Nonlocal dependencies like topicalization, right extraposition, tell us that *trees* are not expressive enough. We need a way to connect an extraposed constituent with its syntactic resp. semantic head. This can be done either by introducing empty leaf nodes plus a means for node coreference (like in the Penn Treebank) or by admitting crossing edges. In our project, the latter solution has been chosen (Skut et al., 1997), partly for the reason that it is simpler to annotate (no decision on the right place of a trace has to be taken). We call this extension of trees with crossing edges *syntax graphs*. An example is shown in Fig. 1.

In order to discuss the details of the language, we will make reference to the simpler syntax graph in Fig. 2.

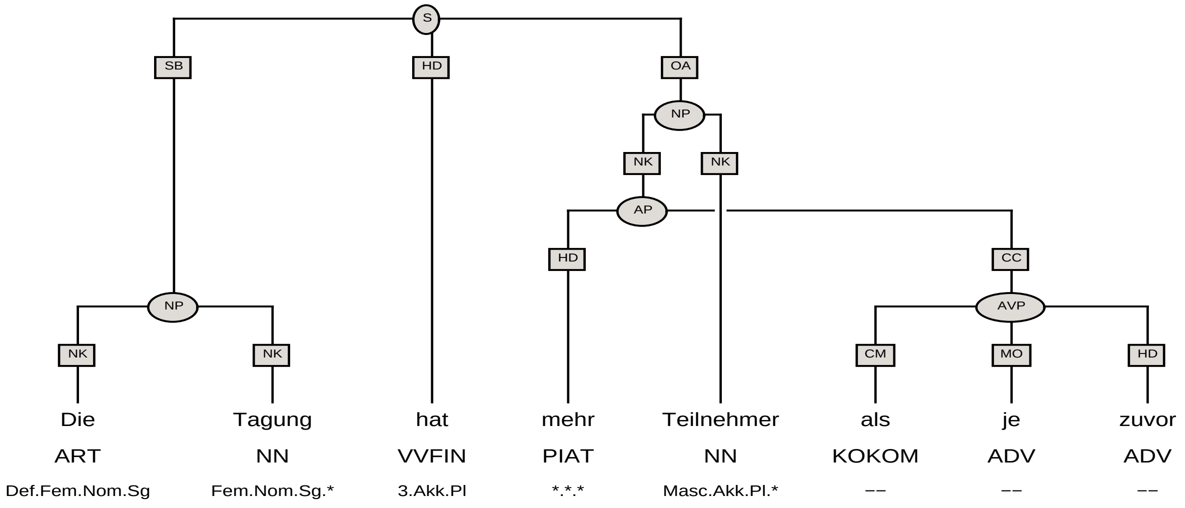

Figure 1: A syntax graph with crossing edges ("the conference has more participants than ever before")

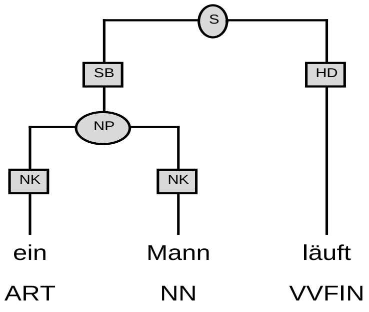

Figure 2: A simple syntax graph ("a man runs")

2.2 Nodes: feature records

Syntactic phrases and lexical entries usually come with a bundle of morphosyntactic information like part-of-speech, case, gender, and number. In computational linguistics, feature structures are used for that purpose. Since we need only a way to represent morphosyntactic information (not syntactic or semantic structures) themselves, we restrict ourselves to *feature records*, i.e. flat feature structures whose feature values are constants. We admit Boolean formulas, for the feature values, as well as for the feature-value pairs themselves.

For example, all proper nouns ("NE") and nouns ("NN") can be retrieved by

```
[pos= "NE" | "NN"]
```

As usual, structural identity can be expressed by the use of logical variables. However, variables must not occur in the scope of negation, since this would introduce the computational overhead of inequality constraints.

The values of a feature with 'infinite' range like `word` or `lemma` can be referred to by regular expressions, e.g. the nouns ("NN") with initial `M` can be retrieved by

```
[word = /^M.*/ & pos="NN"]
```

The `/`-symbols mark a regular expression.

2.3 Node relations

Since graphs are two-dimensional objects, we need one basic node relation for each dimension, direct precedence . for the horizontal dimension and direct dominance `>` for the vertical dimension (the precedence of two inner nodes is defined as the precedence of their leftmost terminal successors (Lezius and König, 2000a)) Some convenient derived node relations are the following:

`>*` dominance (minimum path length 1)

`>n` dominance in n steps ($n > 0$)

`>m,n` dominance between m and n steps ($0 < m < n$)

`>@l` leftmost terminal successor
('left corner')

`>@r` rightmost terminal successor
('right corner')

`.*` precedence (minimum number of intervals: 1)

`.n` precedence with n intervals ($n > 0$)

`.m,n` precedence between m and n intervals ($0 < m < n$)

`$` siblings

`$.*` siblings with precedence

2.4 Graph descriptions

We admit restricted Boolean expressions over node relations, i.e. conjunction and disjunction, but no negation. For example, the queries

```
#n1:[word="ein" & pos="ART"] &
#n2:[word="Mann" & pos="NN"] &
#n1 . #n2
```

and

```
#n1:[cat="NP"] >"NK" [pos="ART"]
& #n1 >"NK" [word="Mann"]
```

are both satisfied by the NP-constituent in Fig. 2. `#n1`, `#n2` are variables. The symbol `"NK"` is an edge label. Edges can be labelled in order to indicate the syntactic relation between two nodes.

2.5 Types

For the purpose of conceptual clarity, the user can define type hierarchies. 'Subtypes' may also be constants e.g. like in the case of part-of-speech symbols. Here is an excerpt from the type hierarchy for the STTS tagset:

```
nominal := noun,properNoun,pronoun.
noun := "NN".
properNoun := "NE".
pronoun :=
  "PPPER","PPOS","PRELS", ... .
```

This hierarchy can be used to formulate queries in a more concise manner:

```
[pos=nominal] .* [pos="VVFIN"]
```

2.6 Templates

E.g. for a concrete lexicon acquisition task, one might have to define a collection of interdependent, complex queries. In order to keep the resulting code tractable and reusable, queries can be organised into templates (or macros). Templates can take logical variables as arguments and may refer to other templates, as long as there is no (embedded) self-reference. Logically, templates are off-line-compilable Horn formula.

Here are some examples for template definitions. A simple notion of `VerbPhrase` is being defined with reference to a notion of `PrepPhrase`.

```
PrepPhrase( #n0:[cat="PP"]
              > #n1:[pos="APPR"] &
            #n0
              > #n2:[pos="NE"] &
            #n1.#n2 ) ;

VerbPhrase( #n0:[cat="VP"]
              > #n1:[pos="VVFIN"] &
            #n0 > #n2 &
            #n1.#n2 ) <-
   PrepPhrase(#n2) ;
```

3 The Corpus Annotation Language

3.1 Corpus annotation vs. queries

Actually, the query language is rather a *description language* which can be used also for encoding the syntactic annotation of a corpus. In the current project, a syntactically disambiguated corpus is being produced. This means, that, for corpus annotation, only a sublanguage of the proposed language is admissible with the following restrictions:

- The graph constraints may only include the basic node relations (`>`, `.`).
- The only logical connective on all structural levels is the conjunction operator `&`.
- Regular expressions are *not* admitted.
- Types and templates are *not* admitted.

The automatically generated corpus annotation code (generated from the output of the graphical annotation interface) for Fig. 2 looks as follows, with some additional markup for ease of processing.

```
<sentence id="1" root="5">
"1":[word="ein" & pos="ART"] &
"2":[word="Mann" & pos="NN"] &
"3":[word="läuft" & pos="VVFIN"]
"4":[cat="NP"] &
"5":[cat="S"] &
("1" . "2") & ("2" . "3") &
("5" >"SB" "4") & ("5" >"HD" "3") &
("4" >"NK" "1") & ("4" >"NK" "2")
```

3.2 An XML representation

When designing the architecture of our system, we had to deal with the problem of various different formats for the representation of syntactically annotated corpora: Penn Treebank, NeGra (Skut et al., 1997), Tipster, Susanne, several formats for chunked texts and the proposed description language. Thus, we have developed an XML based format which guarantees maximum portability (Mengel and Lezius, 2000). An online conversion tool (NeGra, Penn Treebank → XML) is available on our project homepage.

4 Formal Semantics

Compared to most other corpus description and corpus query languages, our graph description language comes with a formal and a clear-cut operational semantics, which has been described in a technical report (Lezius and König, 2000a). The semantics has been compiled from the corresponding parts of formal semantics of the typed, unification-based grammar formalisms and constraint-based logic programming languages which have been cited above. Due to the fact that the corpus and the query are represented in the same description language, one can define a consequence relation between the corpus and the query. Essentially, the annotated corpus corresponds to a Prolog database, and the corpus query to a Prolog query. A query result is a syntax graph from the corpus.

5 Implementation

One might argue that commercial and research implementations for structurally annotated texts are already available, i.e. XML-retrieval systems, c.f. (LTG, 1999). However, we intend to solve problems which are specific to natural language descriptions: non-embedding (non-tree-like) structural annotations - crossing edges, and, on the long-term, retrieval of co-indexed substructures (co-reference phenomena). A domain-specific implementation of the search engine gives the basis for optimizations wrt. linguistic applications (Lezius and König, 2000b).

Before queries can be evaluated on a new corpus (encoded in the NeGra, Penn Treebank or XML format), a preprocessing tool has to convert it into the format of the description language. Subsequently, the corpus is indexed in order to guarantee efficient lookups during the query evaluation. The query processor to date is capable of evaluating basic queries (cf. Sect. 2.2-2.4). To support all popular platforms, the tool is implemented in Java. There is a servlet available on the project web page which illustrates the current stage of the implementation.

Conclusion

Syntactic corpus annotations, complex corpus queries and computational grammars have one common point: they are descriptions of natural language grammars. Our claim is that corpus query languages should be close to traditional grammar formalisms in order to make complicated information extraction tasks easier to encode. The level of processing efficiency of scripting languages can still be reached if one restricts oneself to 'off-line' compilable language elements only.

References

Franz Beil, Glenn Carroll, Detlef Prescher, Stefan Riezler, and Mats Rooth. 1999. Inside-outside estimation of a lexicalized pcfg for german. In *Proceedings of the 37th Annual Meeting of the ACL*, Maryland.

Steven Bird, Peter Buneman, and Tan Wang-Chiew. 2000. Towards a query language for annotation graphs. In *Proceedings of the LREC 2000*, Athens, Greece.

Bob Carpenter. 1992. *The Logic of Typed Feature Structures.* Tracts in Theoretical Computer Science. Cambridge University Press, Cambridge.

Ann Copestake, 1999. *The (new) LKB system.* www-csli.stanford.edu, /~aac/doc5-2.pdf.

Jochen Dörre and Michael Dorna. 1993. CUF - a formalism for linguistic knowledge representation. Deliverable R.1.2A, DYANA 2, August.

Jochen Dörre, Dov M. Gabbay, and Esther König. 1996. Fibred semantics for feature-based grammar logic. *Journal of Logic, Language, and Information. Special Issue on Language and Proof Theory*, 5:387–422.

Denys Duchier and Joachim Niehren. 1999. Solving dominance constraints with finite set constraint programming. Technical report, Universität des Saarlandes, Programming Systems Lab.

Martin Emele and Rémi Zajac. 1990. A fixed-point semantics for feature type systems. In *Proceedings of the 2nd International Workshop on Conditional and Typed Rewriting Systems*, Montreal, Canada.

Markus Höhfeld and Gert Smolka. 1988. Definite relations over constraint languages. LILOG-Report 53, IBM Deutschland, Stuttgart, Baden-Württemberg, October.

Wolfgang Lezius and Esther König. 2000a. The TIGER language - a description language for syntax graphs. Internal report, IMS, University of Stuttgart.

Wolfgang Lezius and Esther König. 2000b. Towards a search engine for syntactically annotated corpora. In *Proceedings of the KONVENS 2000*, Ilmenau, Germany.

LTG Language Technology Group, Edinburgh, 1999. *LT XML version 1.1. User documentation and reference guide.* `www.ltg.ed.ac.uk`, `software/xml`.

Mitchell Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics.*

Andreas Mengel and Wolfgang Lezius. 2000. An XML-based representation format for syntactically annotated corpora. In *Proceedings of the LREC 2000*, Athens, Greece.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP)*, Washington, D.C., March.

Wojciech Skut, Thorsten Brants, Brigitte Krenn, and Hans Uszkoreit. 1998. A linguistically interpreted corpus of german newspaper text. In *ESSLI 1998, Workshop on Recent Advances in Corpus Annotation.*

UCL University College London, 2000. *ICE (International Corpus of English).*